Modeling Buffer Occupancy for a TCP Connection

Yishen Sun, C. C. Lee, Randall Berry and A. H. Haddad
Department of Electrical and Computer Engineering, Northwestern University
Evanston, IL 60208, U.S.A.
Email: {yishen, cclee, rberry, ahaddad}@ece.northwestern.edu

***Abstract*[1] -- In this paper we analyze the buffer occupancy dynamics of a TCP connection for a single-hop system. Our focus is on a single connection experiencing either no losses or one loss. Due to the fact that the majority of TCP flows are very short, we consider the recovery and post recovery stage of a single loss during the slow-start phase. Both TCP Reno and TCP Tahoe implementations are considered, and the results are compared. The model provides some insights into the buffer dynamics statistics and TCP performance parameters. The correctness of the analysis is justified through the OPNET simulation.**

I. Introduction

The dominant congestion control protocol used in the Internet today is the Transport Control Protocol (TCP). A number of analytical models have been proposed in recent years to characterize TCP behavior in terms of latency and throughput (e.g., [1], [2], [3], [4]). In this paper, we consider analytical and graphical models for the buffer occupancy in a router with TCP traffic. A comprehensive understanding of the TCP-related buffer occupancy dynamics is essential for the router to deliver reliable per-flow or per-subscriber Quality of Service (QoS) in an access network. The efficient sizing and management of buffers should achieve a good balance between available resources and QoS requirements, such as the dropping ratio. Moreover, when fair queueing is used, the per-flow buffer occupancy will influence the average throughput experienced by each flow. Since most TCP flows in the Internet are very short [5], they often spend their entire lifetime in TCP's slow-start mode, without suffering any loss or with only a single loss. Therefore, we consider the buffer dynamics with no losses or a single loss during the slow-start phase. For simplicity, we focus on the buffer evolution at the router of a single-hop system with one connection. Extensions to multiple hops and multiple TCP connections are topics of the future work.

The rest of the paper is organized as follows. Section II describes the network model and the assumptions made. In Section III, we study the queue growth dynamics during the slow-start phase, the congestion avoidance phase and the saturation phase respectively, assuming no losses. The scenario in which the receiver deploys the delayed acknowledgement (ACK) scheme is discussed at the end of this section as well. Section IV analyzes the impact of a single packet loss on the buffer occupancy. The OPNET simulation results are given in Section V, and Section VI concludes the paper.

II. Single Hop Model

In the single-hop system we study here, the TCP source connects to the router through a link of capacity C_1, and the router forwards packets to the destination through a link with bandwidth C_2. Note that C_2 may represent the real link capacity, or its value may reflect the share of bandwidth allocated to a specific connection if some scheduling algorithm is implemented at the router for QoS considerations. Let RTT_1 denote the round-trip delay between the sender and the router, and RTT_2 denote the round-trip delay between the router and the receiver. Here, the corresponding propagation delays, processing delays and transmission delays are included in RTT_1 and RTT_2, but queueing delays will be treated separately. We assume that C_1 is much greater than C_2, so the router buffer will be built up immediately by the bursty traffic arrivals. The queue length we get under this assumption can be viewed as the worst case analysis, and it is the upper bound for other cases where the ratio between C_1 and C_2 is not very large. This is because of the fact that the larger the difference between C_1 and C_2, the faster the queue grows at the router.

Similar to [1], [2], [3], [4], we assume that the sender transmits constant sized segments as fast as its congestion window allows and that the receiver advertises a consistent flow control window, W_{max}, throughout the TCP session. The lifetime of a TCP session is divided into "rounds" as described in [2]. A round begins with the transmission of a window of packets and ends on the receipt of an ACK for one of these packets. We assume that the time to transmit all the packets is much smaller than the duration of the round ($\approx RTT_1+RTT_2$) and that the duration of the round is independent of the window size. We will use "packet" and "segment" interchangeably in this paper.

In the following discussion, RTT_1 is assumed to be negligible compared to RTT_2. Thus packets sent from the source can be viewed as arriving at the router buffer

[1] This work was supported by the Motorola Center for Communications at Northwestern University

immediately. However, this assumption is for the convenience of explanation only, and the conclusion can be easily generalized to other values of RTT_1 as long as the staring time of each round is shifted properly. In addition, the models of Section III and IV are valid for the scenario when the destination acknowledges every packet received. Analysis can be done in a similar way for the system with no loss where the delayed ACK scheme is used at the end user, and the main results are included in Section III.E.

III. Buffer Occupancy Analysis with No Loss

Let L, W_k and $B(t)$ denote the segment size, the congestion window size by the end of the kth round, and the router buffer occupancy, respectively. Here $B(t)$ and W_k are measured in units of segments, and a segment is counted in $B(t)$ as long as it has not been completely served. Also, let t_k denote the starting time of the kth round, i.e., $t_k= (k-1)\ RTT_2$. In this section we analyze the buffer dynamics during round k assuming no segments are lost.

A. The Slow-Start Phase

If the session is in the slow-start phase, then $W_k = 2^{k-1}$ in the kth round. For $k = 1$, the buffer occupancy evolves as shown in Figure 1(a), i.e., $B(t) = 1$ for $t \in [t_1, t_1+\tau]$ and $B(t) = 0$ for the rest of the round, where $\tau = L / C_2$. The buffer dynamics for $k > 1$ are depicted in Figure 1(b). Notice that there are W_{k-1} upward jumps in Figure 1(b), each of which corresponds to the arrival of an ACK from the previous round. This is because the congestion window size increases by one for every ACK received during slow-start. This results in two packets being transmitted and joining the buffer. However, one packet is also removed from the buffer. The maximum buffer occupancy of round k is $W_{k-1}+1$. Note that we are assuming here the destination sends an ACK immediately after every segment is received. The delayed ACK scheme can be modeled in a similar approach as shown in Section III.E.

B. The Congestion Avoidance Phase

If the session is in the congestion avoidance phase, then $W_k = W_{k-1}+1$ in the kth round. The resulting buffer dynamics are shown in Figure 1(c). Since the congestion window size increases by one after each round instead of each ACK, the packets' arrival to the router buffer is less bursty during congestion avoidance compared to slow-start. The maximum buffer occupancy of a round is 2.

C. The Saturation Phase

If the session is in the saturation phase, then in the kth round $W_k= W_{max}$. The buffer dynamics are shown in Figure 1(d). The maximum buffer occupancy of a round is 1.

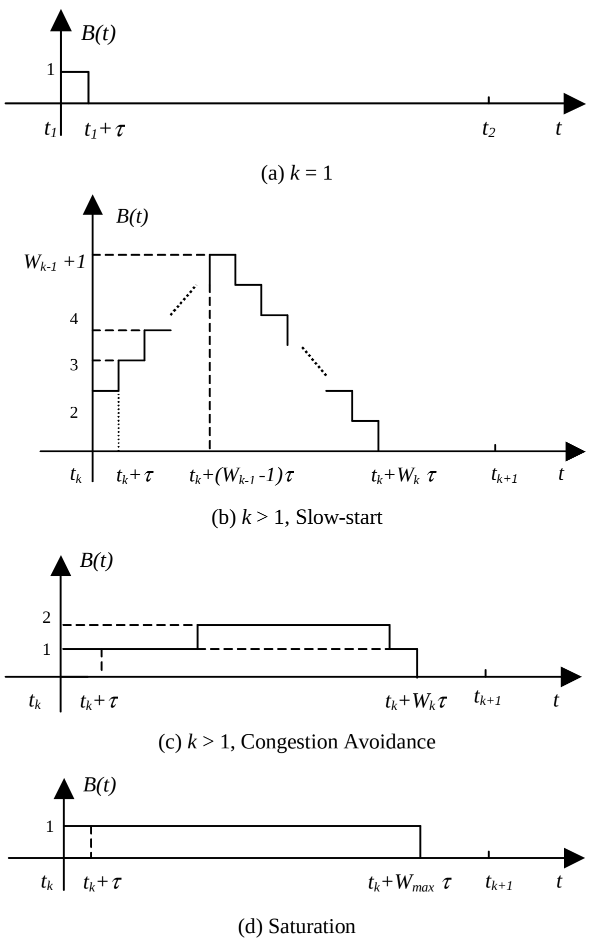


Figure 1 Router Buffer Occupancy during Round k with No Losses

D. Buffer Dynamics Statistics

The analysis above provides some insights into the buffer dynamics statistics in terms of rounds, and TCP performance parameters of a finite size file transfer can be derived as well.

Two statistics of general interest are the maximum buffer occupancy which determines the worst case delay, and the average buffer occupancy which affects the average waiting time at the router.

Let $B_M(k)$ and $B_{av}(k)$ denote the maximum and the average buffer level of the kth round, respectively. Assume the congestion threshold *ssthresh* is known. Here $B_M(k)$, $B_{av}(k)$ and *ssthresh* are all measured in units of segments. Based on the model derived and assumed that W_{max} is large enough, $B_M(k)$ and $B_{av}(k)$ can be expressed as follows:

$$B_M(k) = \begin{cases} 2^{k-2}+1 & ,if\ k \le k_0; \\ ssthresh - 2^{k_0-1}+1 & ,if\ k = k_0+1; \\ 2 & ,if\ k > k_0+1. \end{cases} \qquad (1)$$

where $k_0 = \lfloor \log_2(ssthresh) \rfloor + 1$.

$$B_{av}(k)=\begin{cases}\left(2^{k-2}+1\right)^2\cdot\left(\tau/RTT\right) & ,if\ k\le k_0;\\ B_0\cdot\left(\tau/RTT\right) & ,if\ k=k_0+1;\\ (k-k_0+2^{k_0})\cdot\left(\tau/RTT\right) & ,if\ k>k_0+1.\end{cases}\qquad(2)$$

where

$$B_0=\left(B_M(k_0+1)-1\right)B_M(k_0+1)+B_M(k_0+1)\left(2^{k_0}-ssthresh\right).$$

Figure 2 shows the values of $B_M(k)$ and $B_{av}(k)$ as functions of the round index k. The congestion window size of round k, W_k, is also included in the figure for the comparison purpose. Since the default value of *ssthresh* is 65536 bytes in many systems and the typical TCP segment size is 1460bytes, 44 ($\approx$ 65535/1460) is chosen as the value of *ssthresh* for this plot. In addition, the ratio of *RTT* to τ is set to 25, so that $B_M(k)$ and $B_{av}(k)$ can fit into the figure properly. The value of RTT/τ will not matter as long as the assumption that "the time to transmit all the packets is much smaller than the duration of the round" is satisfied.

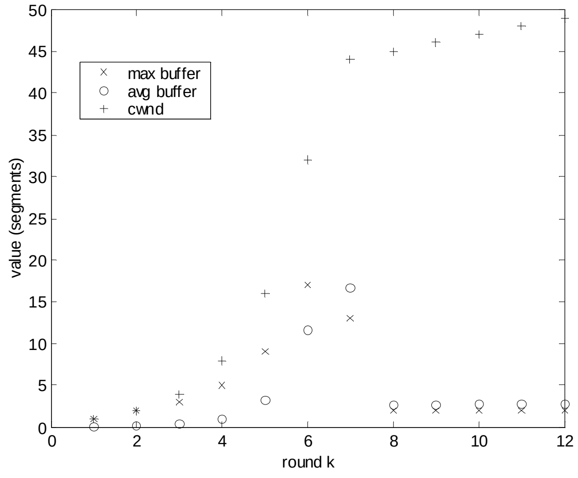


Figure 2 Maximum and Average Buffer Occupancy Per Round

Note that when *ssthresh* = 44, $k_0 = 6$. This implies that the switch from slow-start to congestion avoidance takes place in round 7. It can be seen from Figure 2 that the maximum and the average buffer occupancy of the previous seven rounds are higher than other rounds, while the congestion window size is relatively small, which means fewer packets are delivered. These parameters justify the fact that the traffic arrival during slow-start is more busty than in the congestion avoidance phase.

Equation (1) and (2) can be utilized further to get other results, for example, to calculate the maximum and average buffer occupancy during the lifetime of a finite-size file transfer.

E. Delayed ACK Scheme

In the previous analysis, we assumed the receiver acknowledges every packet it receives. Another widely deployed option for the destination user is to implement the delayed ACK scheme. With delayed ACKs, the receiver sends one ACK for every two packets that it receives or if the delayed ACK timer expires. With this modification, the increase speed of the congestion window is slowed down.

The router buffer occupancy of a TCP connection with delayed ACKs can be modeled following a similar approach. However, the analysis is complicated by the existence of the ACK timer. Some basic results are obtained with the assumption that a round is long enough so that no ACK acknowledges two packets from different rounds.

The congestion window size by the end of the k^{th} round, W_k, is the sequence characterized by

$$W_k=\begin{cases}1 & ,k=1;\\ W_{k-1}+\left\lceil\dfrac{W_{k-1}}{2}\right\rceil & ,k>1\ \text{and in slow-start;}\\ W_{k-2}+1 & ,k>1\ \text{and in congestion avoidance.}\end{cases}$$

The maximum buffer occupancy of the k^{th} round, $B_M(k)$, equals 1, 2, 3, 3 and 3, respectively for the first five rounds. For $k>5$,

$$B_M(k)=\begin{cases}3+\left\lfloor\dfrac{3B_M(k-1)-7}{2}\right\rfloor & ,if\ \text{in slow-start;}\\ 3 & ,if\ \text{in congstion avoidance.}\end{cases}$$

IV. Buffer Occupancy Analysis with a Single Loss

In this section, we investigate how a single segment loss impacts the buffer occupancy at the router. Since the packets arrival of the slow-start phase is more busty than that of the congestion avoidance phase, we will focus on the segment loss occurred in slow-start.

Suppose the i^{th} packet of the k^{th} round is lost. To simplify our discussion, we assume that i is an even number, $i=2j$ and $j<W_{k-1}$. This assumption can be generalized, and the analysis for odd number i or for $j=W_{k-1}$ case can be done using the same approach. The buffer occupancy dynamics will be slightly different without the above assumption, but the key properties mentioned at the end of the section are preserved.

After a packet is lost, a sequence of events takes place at the sender to recover from the detected loss through TCP's fast retransmit and/or fast recovery algorithm [6]. We name this period the "recovery stage". After the successful retransmission of the lost segment, the TCP source enters the congestion avoidance or slow-start phase; we call this the "post-recovery stage".

In Section IV.A we assume the sender is using a congestion control algorithm from the TCP Reno family, which is currently the most widely implemented TCP version. The results for the Tahoe version are included in Section IV.B. Section IV.C compares the conclusions of both versions.

A. Reno

TCP Reno implements both fast retransmit and fast recovery algorithm.

The outline of the important events during the recovery stage is listed in Table 1, where *W(t)* and *ssthresh* is the congestion window size and the slow-start threshold at time *t* (measured in segments), and E_n is the notation for Event n.

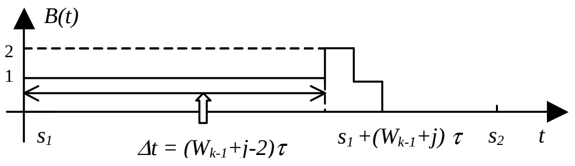
E_1:	$T= t_k+(j-1)\tau$. The $(2j)^{th}$ segment of this round is lost. $B(t)=j$, $W(t)= W_{k-1}+j$, *ssthresh*= default value;
E_2:	$T= t_{k+1}+(2j-2)\tau$. The sender receives the first ACK which asks for the lost packet. $B(t)=2j$, $W(t)=W_k+(2j-1)$;
E_3:	$T= t_{k+1}+(2j)\tau$. The third duplicate ACK is received by the source, and the lost packet is retransmitted. $B(t)=2j-2$. Adjust $W(t)$ from $W_k+(2j-1)$ to $W_{k-1}+(j+2)$, and *ssthresh* from the default value to $W_{k-1}+(j-1)$;
E_4:	$t_{k+1}+(2j)\tau < t < t_{k+1}+(W_k-1)\tau$ or $t_{k+2} < t < t_{k+2}+(3j-1-W_{k-1})\tau$. The size of the segments in flight is bigger than $W(t)$. No new packets are released. The source keeps receiving duplicate ACKs and increasing $W(t)$ by 1 at every incoming ACK. $B(t)$ decreases to 0 gradually;
E_5:	$t_{k+2}+(3j-1-W_{k-1})\tau \leq t < t_{k+2}+(4j-2)\tau$. The size of the segments in flight is equal to $W(t)$. The sender releases one new packet for every duplicate ACK received, and $W(t)$ is increased by 1 as well. $B(t)=1$;
E_6:	$T= t_{k+2}+(4j-2)\tau$. The ACK of the successful retransmission arrives at the source, which acknowledges all intermediate segments and ends the fast retransmit and fast recovery algorithm. $W(t)=W_k+(2j-1)=ssthresh$, so the sender enters the congestion avoidance phase. $B(t)=1$;
E_7:	$T= t_{k+2}+(W_{k-1}+3j-2)\tau$. $W(t)=ssthresh=W_k+(2j-1)$. The post recovery stage begins.

Table 1 Event List of the Recovery Stage for Reno

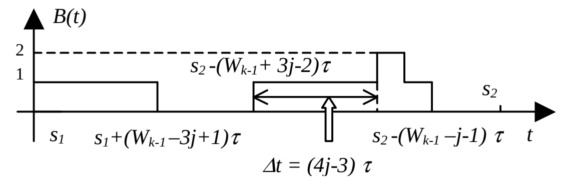
Notice that the time intervals specified for E_4 and E_5 in Table 1 are valid assuming that $j>(W_{k-1}+1)/3$. If $j\leq (W_{k-1}+1)/3$, the time range of E_4 should be changed to $t_{k+1}+(2j)\tau < t < t_{k+1}+(W_{k-1}+3j-2)\tau$, and the time range of E_5 should be changed to $t_{k+1}+(W_{k-1}+3j-2)\tau \leq t < t_{k+1}+(W_k-1)\tau$ or $t_{k+2} < t < t_{k+2}+(4j-2)\tau$.

Figure 3(a) plots the queue growth during the recovery stage after the packet loss for $j>(W_{k-1}+1)/3$. Figure 3(b) is for $j\leq (W_{k-1}+1)/3$. Figure 3 is attached at the very end of this paper.

The post-recovery stage starts from the moment $s_1= t_{k+3}+(3j-1-W_{k-1})\tau$ for $j>(W_{k-1}+1)/3$ or $s_1 = t_{k+2}+ (W_{k-1}+3j-2)\tau$ for $j\leq (W_{k-1}+1)/3$. The sender is in congestion avoidance phase from the beginning of the post recovery stage. The buffer occupancy of the first round of this stage is shown in Figure 4(a) and (b) for both cases respectively. The buffer dynamics in the following rounds are very similar to the first round with the difference that the span of Δt, as marked in the figure, is increased by τ after each round.



(a) Post Recovery stage, Reno, $j > (W_{k-1}+1)/3$



(b) Post Recovery stage, Reno, $j\leq (W_{k-1}+1)/3$

Figure 4 Buffer Occupancy of the First Round of the Post Recovery Stage for the sender implementing TCP Reno

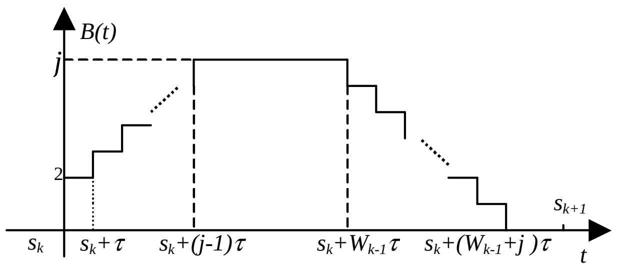
An interesting observation is that when the segment loss occurs, the buffer occupancy is *j* segments, while during the entire recovery and post recovery stage, *B(t)* never exceeds the maximum of W_{k-1} and $2j$. Since this is also true for the Tahoe implementation, it is summarized as the following proposition. The verification for Tahoe scenario can be found in Section IV.B. The potential application of this property is outlined in Section IV.C.

Proposition 1: If a connection experiences a single segment loss during the slow-start phase, the maximum buffer occupancy after the loss will be no greater than either twice the buffer level when loss occurs or twice the maximum buffer occupancy before the loss.

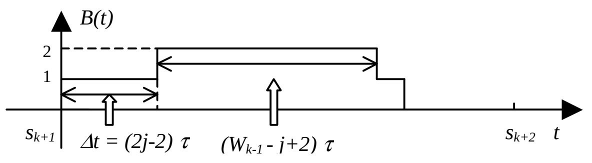
B. Tahoe

TCP Tahoe implements the fast retransmit algorithm, but no fast recovery scheme. Therefore the queue growth of the recovery stage before $t= t_{k+1}+(2j)\tau$, the time when the packet loss is detected, is the same as Reno, which implies that Event 1 to 3 in Table 1 also apply to Tahoe scenario. However, because of the lack of the fast recovery mechanism, no new packet is released from the sender after detecting the packet loss until the ACK of the retransmitted packet has returned successfully at $t= t_{k+2}+(4j-2)\tau$. The buffer is emptied gradually during that interval.

The post recovery stage starts from $s_2= t_{k+2}+(4j-2)\tau$, and the buffer growth begins from the second round of the slow-start phase as specified in Section III.A. If we renumber the beginning round of the post recovery stage as round 2, then the buffer occupancy of round *k* is shown in Figure 5(a). Round *k* is of particular interest because the sender switches from slow-start to congestion avoidance during this period of time, and it is the round where the buffer level reaches its maximum of the post recovery stage. Notice that as long as the buffer occupancy hits *j*, which is the buffer level when the previous loss occurred, the sender enters the congestion avoidance phase, and the queue never exceeds j thereafter. The buffer occupancy of round $k+1$ of the post recovery stage is shown in Figure 5(b). The queue dynamics in the following rounds are very similar to the $(k+1)$ round with the difference that the span of Δt, as marked in the figure, is increased by τ after each round.



(a) Post Recovery Stage, Tahoe, round *k*



(b) Post Recovery Stage, Tahoe, round $(k+1)$

Figure 5 Buffer Occupancy of the Post Recovery Stage for the sender implementing TCP Tahoe

It is easy to verify that during the recovery stage the maximum queue length of the system with the Tahoe sender is no greater than that with the Reno user. Together with the fact that the maximum buffer level of the post recovery stage is j for Tahoe, it is obvious that the proposition 1 stated at the end of Section IV.A also holds for the TCP Tahoe family.

C. Comparison of Reno and Tahoe

The different behavior of Tahoe vs. Reno is reflected in the buffer occupancy change after the lost packet is detected. The main discrepancy shows in two aspects.

In the recovery stage, no new packet is released from the sender in the Tahoe scenario after detecting the packet loss until the ACK of the retransmitted packet has returned successfully at $t= t_{k+2}+(4j-2)\tau$, while according to the fast recovery algorithm, $(W_{k-1}+j-1)$ packets are transmitted during that interval for the Reno case.

In the post recovery stage the sender using Tahoe starts from the slow-start phase and increases gradually into the congestion avoidance phase. However, the Reno user enters the congestion avoidance phase directly from the beginning of the stage, so more packets are sent and the traffic arrival is less bursty.

A quantitative example is given in Figure 6 to compare the maximum buffer occupancy and the number of packets sent in each round of both versions. In this example we assume the 14th packet of the 5th round is lost. Round 0 refers to the time interval between $t=t_{k+1}+(2j)\tau$ and $t= t_{k+2}+(W_{k-1}+3j-2)\tau$, and the numbering of the round follows the definition of the new round in Section IV.A for the Reno post recovery stage. From the graph it can be seen that Reno is more capable of sending more packets at a lower buffer level than Tahoe.

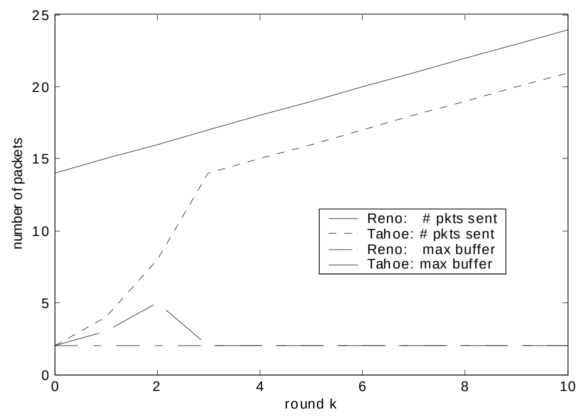


Figure 6 Comparison of Reno and Tahoe performance after a packet loss

Despite the fast recovery algorithm difference between TCP Reno and Tahoe, we also find from the above analysis that the buffer dynamics in both versions have several features in common. During the post recovery stage, $B(t)$ exhibits a non-bursty self-repetitive pattern. More importantly, as stated in the proposition 1, the buffer occupancy never exceeds the maximum of W_{k-1} and $2j$, where W_{k-1} +2 is twice the maximum buffer level of previous rounds, and $2j$ is twice the buffer size when the loss occurs. This holds for both the recovery stage and the post recovery stage. If j satisfies $W_{k-2}+1 < j < W_{k-1}$, then it is always true that $2j > W_{k-1}$. Notice that $W_{k-2}+1$ is the maximum queue length of all previous $(k-1)$ rounds, this property implies a potential queue management scheme: choose a virtual buffer limit j such that $W_{k-2}+1 < j < W_{k-1}$, and mark or drop the first packet which reaches this level, with no further marking or dropping. Then during the following transmission, the actual queue length is bounded by $2j$. If j is no greater than the half of the real buffer limit, then overflow can be avoided if marking packets in this way.

V. Simulation

An OPNET simulation model has been set up to verify the buffer occupancy model we proposed. The system structure is the same as the single-hop system described in Section II, with C_1 using T3 link, C_2 using DS1 link, and RTT_2 is approximately 0.5sec. The simulation results are very consistent with our queue length growth model. As an example, Figure 7 shows the evolution of the router buffer when the connection experiences a single loss and the sender implements TCP Tahoe. It can be seen clearly from the statistics that the post recovery stage contains both slow-start phase and congestion avoidance phase.

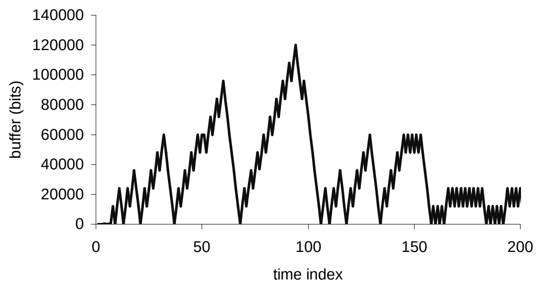


Figure 7 Buffer Occupancy for a Connection with Single Loss, Tahoe

VI. Conclusion and discussion

In this paper we analyze the buffer occupancy dynamics of a one-hop system. Our focus is on a single connection experiencing one loss or no losses. Based on the model we proposed, some buffer dynamics statistics are studied and the results may provide some insights into the TCP performance parameters of a finite size file transfer, such as average buffer occupancy and the idling time during the entire connection. This information may help addressing the buffer sizing issue at the edge router as well. It can be seen from the model that the active time ratio of a session is related to the allocated bandwidth share C_2, which suggests the interaction between the number of active flows, the total number of flows and the scheduling algorithm. Another observation is that under certain constraints the maximum buffer occupancy during the recovery and the post recovery stage is bounded by twice the buffer size when the loss occurs. A generic queue management scheme might be able to exploit this to prevent buffer overflow if we either mark or drop a segment when the buffer is half full. Modeling buffer occupancy for multiple connections is a topic for future work.

References

[1] N. Cardwell, S. Savage and T. Anderson, “Modeling TCP latency,” *Proceedings of IEEE INFOCOM,* Tel Aviv, Israel, March 2000.

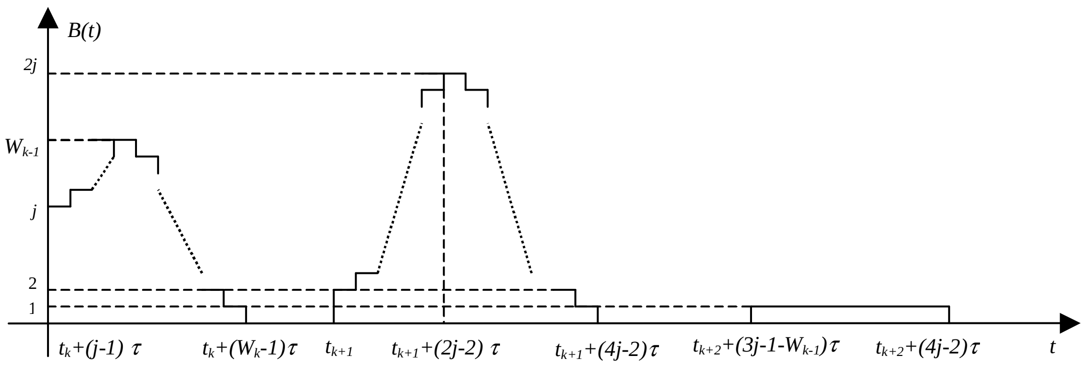
[2] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP Reno performance: A simple model and its empirical validation,” *IEEE/ACM Trans. on Networking,* vol. 8, no. 2, pp. 133-145, April 2000.

[3] B. Sikdar, S. Kalyanaraman and K. S. Vastola, “TCP Reno with Random losses: Latency, Throughput and Sensitivity Analysis,” *Proceedings of IEEE IPCCC,* Phoenix, AZ, April 2001.

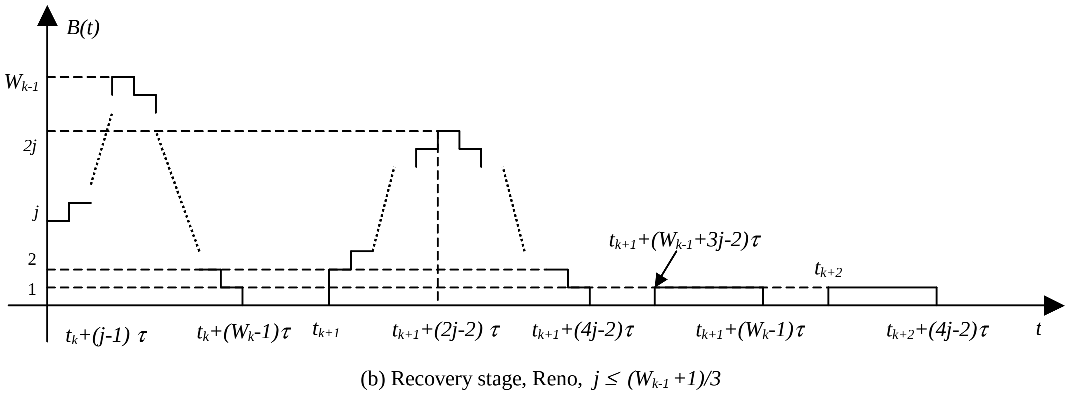
[4] B. Sikdar, S. Kalyanaraman and K. S. Vastola, “Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK,” *IEEE/ACM Trans. on Networking,* vol. 11, no. 6, pp. 959-971, December 2003

[5] K. Thompson, G. J. Miller, and R. Wilder, “Wild-area Internet Traffic Patterns and Characteristics,” *IEEE Network,* 11(6), November 1997

[6] W. R. Stevens, “TCP/IP illustrated volume 1,” Addison Wesley, 1994.



(a) Recovery stage, Reno, $j > (W_{k-1}+1)/3$



(b) Recovery stage, Reno, $j \le (W_{k-1}+1)/3$

Figure 3 Buffer Occupancy of the Recovery Stage for the sender implementing TCP Reno